

Solving the Vlasov equation for one-dimensional models with long range interactions on a GPU

Tarcísio M. Rocha Filho ^{a,1}

^a*Instituto de Física and International Center for Condensed Matter Physics
Universidade de Brasília, CP: 04455, 70919-970 - Brasília, Brazil*

Abstract

We present a GPU parallel implementation of the numeric integration of the Vlasov equation in one spatial dimension based on a second order time-split algorithm with a local modified cubic-spline interpolation. We apply our approach to three different systems with long-range interactions: the Hamiltonian Mean Field, Ring and the self-gravitating sheet models. Speedups and accuracy for each model and different grid resolutions are presented.

Key words: Vlasov equation; Long-range interaction;

1 Introduction

Systems with long-range interactions are particularly important in physics, Coulomb forces being probably the most prominent example. Albeit their relevance, many of its properties are still not well understood. The long-range nature of the interaction leads to some interesting phenomena not observed for short-range interactions, such as the existence of quasi-stationary non-Gaussian states with diverging life-times with the number of particles, negative microcanonical heat capacity, inequivalence of ensembles and non-ergodicity [1,2,3,4,5,6,7]. Examples of systems with long range forces include self-gravitating systems [9], non-neutral plasmas [10,11] and models as the ring model [12,13] Hamiltonian Mean Field (HMF) [14], one-dimensional gravity (infinite uniform density sheets) [15,16,17], Free Electron Laser [18] and plasma single wave models [19], among others. For out of equilibrium situations, many of these studies rely on molecular dynamics simulations, i. e. solving numerically the Hamiltonian equations of motion for the N -particle system. It is also

¹ e-mail: marciano@fis.unb.br

a well known fact that, under suitable conditions, the statistical description of the dynamics of long range interacting systems is equivalent to the Vlasov equation [1,20]. The numerical solution of the Vlasov equation was applied to the HMF model in Ref. [21] and more recently to characterize non-equilibrium phase-transitions in the same model [22,23], although the phase diagram is still open to a closer scrutiny [24].

One-dimensional models are important for a better understanding of many properties of long-range interacting systems. Therefore a fast numeric implementation of the solution of the Vlasov equation is of uttermost value in their investigation. Numerical solutions of the Vlasov equation are obtained either by Particle In Cell (PIC) methods [25], where the distribution function is represented by a collection of macro-particles under the dynamics of the self-consistent mean-field force, or Eulerian methods where the distribution is represented as the density of a non-compressible fluid on a numerical grid [26]. For higher dimensional models, PIC methods are more effective in computational effort, even though its applicability is limited by inherent statistical noise and a poor description of the tails of the distribution. On the other hand, Eulerian methods are limited at higher dimensions by the number of grid points required to accurately represent the distribution function (see [31,32,33] for a comparison of different Eulerian codes). With the rapid increase in computational power and the use of parallel machines Eulerian codes have been implemented up to two spatial dimensions [27,28,29,30].

We present in this paper an implementation in the CUDA framework [34] of a semi-Lagrangian solution method for the Vlasov equation in one dimensional systems with long range interactions, and applications to the ring, HMF and self-gravitating sheet models. In this approach the distribution function is represented on a numerical grid and a times-split algorithm is used to evolve the function by computing the characteristic curves and equating the value of the solution to its value at the foot of the characteristic [23,35,36]. This last step requires an interpolation scheme, and a common choice is to use a cubic spline, which is global on the grid due to the requirement to compute second order derivatives of the distribution function at grid points [37]. As a consequence its parallel implementation is of limited efficiency. An alternative in Ref. [38] is to use a local spline on patches (tiles) in the grid, with the continuity of first derivatives at the borders of each patch. The values of the distribution at the grid points on each patch is stored in shared memory, and all steps are then performed on a patch-by-patch basis. This approach requires communication between processors handling different patches, which can be reduced by suitably restricting the time step [36]. Here we implement a different approach where the interpolation relies on the same form of cubic spline with second order derivatives computed from an eighth order finite difference method.

The structure of the paper is as follows: in section 2 we present the one-dimensional models to which our approach is applied, and section 3 presents and discusses the algorithms implemented in CUDA for the solution of the Vlasov equation. Section 4 presents the results obtained from the implementation of the algorithm to the one-dimensional models, and speedups relative to a serial code. We conclude the paper with some concluding remarks in section 5.

2 One-dimensional models

Since a detailed direct study of real three-dimensional systems with long-range interactions is a very difficult task, some simplified models have been introduced in the literature retaining qualitative features of realistic long-range systems (see [1] and references therein). The Hamiltonian of a one-dimensional model of N identical particles with unit mass can be written as:

$$H = \frac{1}{2} \sum_{i=1}^N p_i^2 + \frac{1}{N} \sum_{i < j=1}^N V_{ij}, \quad (1)$$

where V_{ij} is the potential energy between particles i and j and the factor N^{-1} ensures extensivity of the energy and corresponds to a change of time units. The three different models considered here correspond to different choices for V_{ij} . The Ring model describes a system of N identical particles on a ring or radius R interacting through their gravitational attraction [12,13]. With a choice of units, the interacting potential is given by

$$V_{ij} = -\frac{1}{\sqrt{2}\sqrt{1 - \cos(\theta_i - \theta_j) + \epsilon}}, \quad (2)$$

where θ_i is an angle coordinate specifying the position of particle i on the circle, and ϵ is a (small) softening parameter used to avoid the divergence of the potential at zero distance. For increasing ϵ the ring model tends to the HMF model, with potential [14]:

$$V_{ij} = 1 - \cos(\theta_i - \theta_j). \quad (3)$$

The third system considered is the sheet model formed by N identical infinite self-gravitating parallel sheets of constant mass density [16,17]. The gravitational force between two sheets is therefore constant and they are allowed to cross each other, when the force changes sign. Considering only the motion in

the direction x perpendicular to the sheets, and again with a choice of units, the pair interaction potential is written as:

$$V_{ij} = |x_i - x_j|. \quad (4)$$

The Vlasov equation for those models is thus:

$$\dot{f} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x}v + \frac{\partial f}{\partial v}F(x, t) = 0, \quad (5)$$

where $f = f(x, v, t)$ is the one particle distribution function, v the velocity of the particle, x the position coordinate (θ for the HMF and ring models), and the mean-field force $F(x, t)$:

$$F(x, t) = -\frac{\partial}{\partial x} \int v(x - x')f(x', v', t) dx' dv', \quad (6)$$

with $v(x - x')$ given in eqs. (2–4). For the HMF model the mean-field force can be written as:

$$F(\theta, t) = -\sin(\theta)M_x + \cos(\theta)M_y, \quad (7)$$

where the components of the “magnetization” vector \mathbf{M} are then

$$M_x = \int \cos(\theta)f(\theta, v, t) dx dv, \quad M_y = \int \sin(\theta)f(\theta, v, t) dx dv. \quad (8)$$

This property implies that molecular dynamics simulation times for the HMF model with N particles scale as N instead of N^2 , and is one the reasons why it is so extensively studied.

3 Algorithms and CUDA implementation

The semi-Lagrangian scheme used here is described in References [23,35,36] and can be summarized as follows. The one-particle distribution function is represented in a numerical grid in the one-particle phase space as $f(x_i, p_j, t)$ where x_i and p_j are position and velocity coordinates of the points in the grid on a finite domain $x \in [x_{min}, x_{max}]$ and $p \in [p_{min}, p_{max}]$. The distribution function at time $t + \Delta t$ is obtained numerically by evolving the function back in time and using the invariance of f along the characteristic lines. This backwards evolution is performed using a a time-split method. The mains steps are:

- (1) Backwards time evolution (advection) of f in the spatial direction by a time step $\Delta t/2$ with constant momentum:

$$f^{(I)}(x, p) = f(x - p \Delta t/2, p, t). \quad (9)$$

- (2) Computation of the mean-field force using $f^{(I)}$:

$$F^{(I)}(x) = - \int \frac{\partial}{\partial x} v(x - x') f^{(I)}(x', p') dx' dv'. \quad (10)$$

- (3) Backwards time evolution in the momentum direction by a full time step Δt using $F^{(I)}(x)$:

$$f^{(II)}(x, p) = f(x, p - F^{(I)}(x) \Delta t, t). \quad (11)$$

- (4) And as last step repeat (1):

$$f(x, p, t + \Delta t) = f^{(II)}(x - p \Delta t/2, p, t). \quad (12)$$

The values of the intermediate $f^{(I)}$, $f^{(II)}$ and final distribution functions in steps (1), (3) and (4) at the numerical grid points must be obtained from the known values at the previous step with a cubic spline as interpolation method. For a point with coordinate x , $x_i \leq x \leq x_{i+1}$, the interpolated value for $f(x)$ knowing $f_i = f(x_i)$ and $f_{i+1} = f(x_{i+1})$ is given by [37]:

$$f(x) = \alpha f_i + \beta f_{i+1} + \gamma f_i'' + \delta f_{i+1}'', \quad (13)$$

where

$$\begin{aligned} \alpha &= \frac{x_{i+1} - x}{x_{i+1} - x_i}, \quad \beta = 1 - \alpha, \\ \gamma &= \frac{\alpha^3 - \alpha}{6} (x_{i+1} - x_i)^2, \quad \delta = \frac{\beta^3 - \beta}{6} (x_{i+1} - x_i)^2, \end{aligned} \quad (14)$$

and f_i'' stands for the second derivative of f at x_i . By requiring that first order derivatives computed from eq. (13) are continuum across the boundaries of neighboring intervals, we obtain a tridiagonal system of equations for f_i'' :

$$\left(\frac{f_{i+1}''}{6} + \frac{f_i''}{3} + \frac{f_{i-1}''}{6} \right) \Delta x^2 = f_{i+1} - 2f_i + f_{i-1}, \quad i = 0, \dots, n, \quad (15)$$

with n the number of points in the corresponding direction. Even though useful in a sequential context, a direct efficient parallel implementation of the solution of system (15) is not effective enough.

In order to compute the second order derivatives locally, i. e. involving only a small number of neighbor points, with good accuracy in order not to spoil

the quality of the cubic interpolation an eighth order centered finite difference approximation is used [40]:

$$f''(u_i) = -\frac{1}{560}f(u_{i-4}) + \frac{8}{315}f(u_{i-3}) - \frac{1}{5}f(u_{i-2}) + \frac{8}{5}f(u_{i-1}) - \frac{205}{72}f(u_i) + \frac{8}{5}f(u_{i+1}) - \frac{1}{5}f(u_{i+2}) + \frac{8}{315}f(u_{i+3}) - \frac{1}{560}f(u_{i+4}), \quad (16)$$

where u stands for either the momentum or position variables. Periodic boundary conditions are used both in the spatial and momentum direction. Unphysical effects are avoided by choosing the size of the domain sufficiently large.

The distribution function is represented on an equally spaced grid (x_i, p_j) , with $n_x \times n_p$ points, by a one-dimensional array $f(x_i, p_j) \rightarrow f[j \cdot n_x + i]$, $i = 0, \dots, n_x$ and $j = 0, \dots, n_p$. The second order derivatives are represented similarly. The initial condition array is loaded in global GPU memory and all subsequent operations are performed there. Memory bandwidth is an important issue for efficiency in this memory intensive application, and the algorithm must exploit as much as possible coalesced memory access. Reading and writing in the one-dimensional array in the x direction tends to be coalesced, but not on the p direction. This is an issue when computing the spline coefficients (the second order derivatives) necessary for the interpolation in the spatial advection, but is avoided in the other parts of the advection process. To overcome this difficulty a global transpose of the f array is performed before computing f'' , and then another transpose is performed on the array f'' . As a consequence the same routine is used to compute the derivatives for both spatial and momentum directions. The transpose of a bidimensional array written in the one-dimensional form can be performed efficiently close to full bandwidth in CUDA [39]. Our algorithm is synthesized as:

- (i) Transpose f ;
- (ii) Compute f'' ;
- (iii) Transpose f'' ;
- (iv) Perform a spatial advection (step 1 above) using f'' obtained in step (ii) for the spline interpolation;
- (v) Compute the mean-field force $F^{(I)}$ (step 2 above);
- (vi) Compute f'' ;
- (vii) Perform a momentum advection (step 3) using f'' from step vi;
- (viii) Repeat steps (i–v) (step 4).

The computation of the force in step (v) is implemented by a discretization of eq. (6).

$n_p = n_x$	$\delta e^{(FD)}$	$\delta Norm^{(FD)}$	$\delta e^{(GS)}$	$\delta Norm^{(GS)}$
256	7×10^{-3}	10^{-3}	10^{-2}	10^{-3}
512	7×10^{-4}	10^{-4}	10^{-4}	1.7×10^{-5}
1024	10^{-5}	3×10^{-6}	2×10^{-6}	9×10^{-7}
2048	4×10^{-6}	10^{-8}	3×10^{-6}	10^{-8}

Table 1

Maximum relative error for the energy (δe) and norm ($\delta Norm$) for the Finite Difference (FD) approximation in eq. (16) and Global Spline (GS), with $\Delta = 0.01$.

4 Results and discussions

The simulations presented here were performed on a GTX 560 Ti GPU with 384 cores, 1GB global memory and 1,64 GHz clock speed, and a GTX 590 GPU with 512 cores, 1,5 GB global memory and 1.26 GHz clock (in fact the GTX 590 has two identical devices but only one was used for the runs). The CPU has an i7-2600 processor with 3.4GHz clock and 16GB of RAM. In this section we present and discuss the results of simulations for the three one-dimensional models presented in section 2. All computations performed use double precision.

4.1 Self-gravitating sheet model

As a first test case let us apply our approach to the self-gravitating sheet model with pair interaction potential given by eq. (4), with a (waterbag) constant distribution in an interval as initial condition, i. e:

$$f(x, p, t = 0) = 1/4p_0x_0, \quad \text{if } -x_0 \leq x \leq x_0, \text{ and } -p_0 \leq p \leq p_0. \quad (17)$$

The parameters of the numeric grid are $x_{max} = -x_{min} = 2.0$, $p_{max} = -p_{min} = 2.0$ and $n_p = n_x = 256, 512, 1024, 2048$ (different number of points in each direction can also be used). The waterbag initial condition is chosen with $x_0 = 1.0$ and $p_0 = 0.5$. Two time steps $\Delta = 0.1$ and $\Delta t = 0.01$ were considered to assess numerical errors. Figure 1 shows some snapshots of the time evolution of the distribution function obtained from our code.

Tables 1 and 2 present the relative errors for the energy δe and total norm $\delta Norm$. The accuracy of our approach is similar to the a global spline as describe in [37], for all cases considered. The speedups obtained by comparing our parallel code to a CPU serial version using the same interpolation method are shown in table 3. The speedups grow with n_p and n_x for two main reasons. First not all the latencies of the GPU are covered with a small number of grid

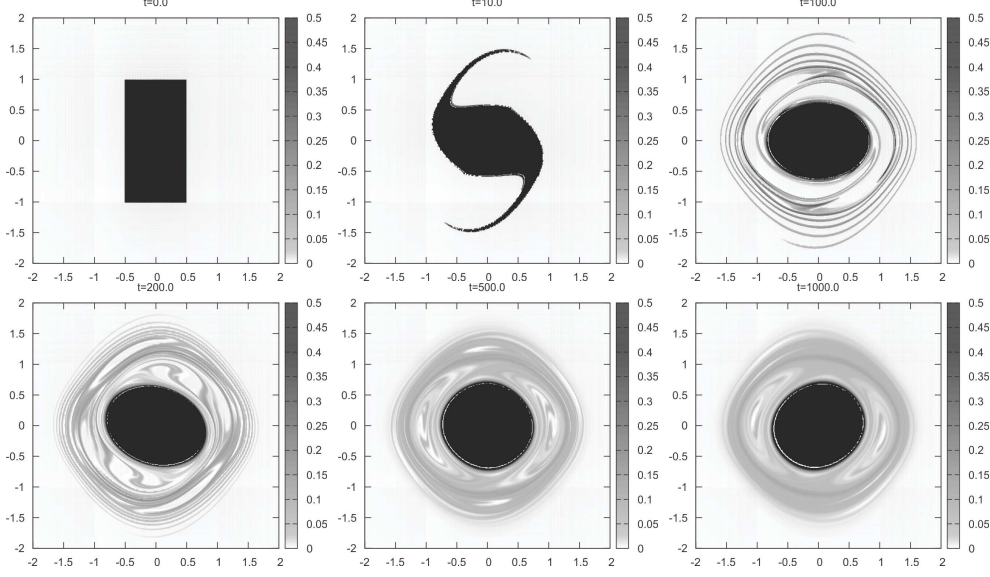


Fig. 1. Snapshots of the distribution function for the self-gravitating sheet model with $x_{max} = -x_{min} = 2.0$, $p_{max} = -p_{min} = 2.0$, $n_p = n_x = 2048$, $\Delta t = 0.01$, a waterbag initial condition with $x_0 = 1.0$ and $p_0 = 0.5$, and $t = 0, 10, 100, 200, 500, 1000$. In each graphic p and x correspond to the horizontal and vertical axis respectively

$n_p = n_x$	$\delta e^{(FD)}$	$\delta Norm^{(FD)}$	$\delta e^{(GS)}$	$\delta Norm^{(GS)}$
256	5×10^{-4}	4×10^{-5}	2×10^{-4}	4×10^{-5}
512	4×10^{-4}	2×10^{-7}	4×10^{-4}	2×10^{-7}
1024	3×10^{-4}	3×10^{-10}	3×10^{-4}	10^{-10}
2048	3×10^{-4}	8×10^{-13}	3×10^{-4}	10^{-14}

Table 2

Same as table 1 with $\Delta t = 0.1$.

$n_p = n_x$	GTX 570 Ti	GTX 590
256	17	20
512	25	31
1024	38	51
2048	51	71

Table 3

Speedups for the self-gravitating sheet model for different grid resolutions.

points. And second, with smaller grid spacings the possibility of coalesced access to global memory is significantly increased.

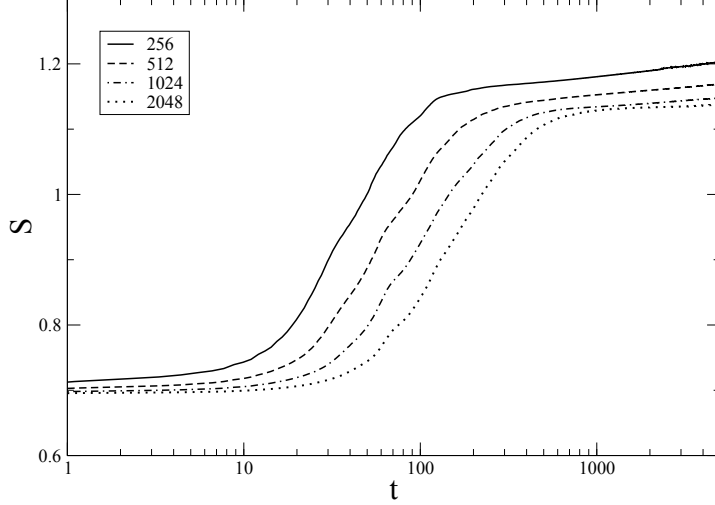


Fig. 2. Entropy for the sheet model for different grid resolutions. The runs are the same as in table 2.

The Vlasov dynamics has an infinite number of invariants, called Casimirs, of the form

$$C[s] = \int s(f(p, x, t)) dp dx. \quad (18)$$

This fact can be used to assess how information on the initial condition is lost due to the finite resolution of the numerical grid and other sources of errors. For this purpose we consider the entropy of the distribution f given by $s(f) = -f \log f$ in eq. (18). Figure 2 shows the time dependence of S for different grid resolutions. Information loss starts to be significant when filamentation is of the order of the grid spacing. These non-Vlasov effects are inherent to Eulerian solvers and must be considered with due care in the numerical solutions of the Vlasov equation [21,41,42,43]. For the highest resolution (2024×2024), there are two plateaus, one at the initial stage, before the formation of filamentation, and another at the final stage, when details smaller than the grid resolutions were lost.

4.1.1 The HMF model

For the HMF model as defined by the pair interaction potential in eq. (3) Molecular Dynamics (MD) simulations scale with the number of particles N . Therefore it is possible to compare results from MD simulations with the solutions of the Vlasov equation, which describes the statistical properties of the particle dynamics in the $N \rightarrow \text{limit}$ [1,20]. As initial condition we consider a waterbag with total energy per particle $e = 0.7$ and average magnetization $M = \sqrt{M_x^2 + M_y^2} = 0.8$, with $M_x \equiv \langle \cos(\theta) \rangle$, $M_y \equiv \langle \sin(\theta) \rangle$. This corresponds to a uniform distribution in the interval $0 \leq \theta \leq 2.262$ and $-1.766 \leq p \leq$

$n_p = n_x$	GTX 570 Ti	GTX 590
256	22	24
512	29	35
1024	35	37
2048	39	53

Table 4

Speedups for the HMF model for different grid resolutions.

1.766. All integrations were performed with $\Delta t = 0.1$, a spatial grid $0 \leq \theta < 2\pi$ and momentum grid $-p_{max} < p < p_{max}$ with $p_{max} = 3.531$ and $n_p = n_\theta = 256, 512, 1024, 2048$. Snapshots of the time evolution of the distribution are shown in Fig. 3 with a strong filamentation already present at $t = 50$.

Figure 4 shows the graphic of the potential energy obtained from a MD simulation with $N = 20,000,000$ particle using a symplectic integrator with time step $\Delta t = 0.1$ [44], and the same curve obtained from our code with a numerical grid with $n_p = n_\theta = 2048$ points. Both simulations are in very good agreement up to roughly $t \approx 70.0$, after which the details of small fluctuations differ. This is due to the finite number of particles in the MD simulation and the strong formation of filaments by the natural evolution of the distribution function down to scales of the size of the grid spacing. Nevertheless the asymptotic behavior is the same in both cases. The entropy for different grid resolutions is shown in Fig. 5. Analogously to the sheet model, it has two plateaus, one before indentations scale reaches grid resolution, and a final plateau after the distribution is coarse grained. The speedups for the HMF model are shown in table 4, and are somewhat smaller than those for the sheet model. This comes from the fact that the serial code is well optimized by using eq. (7). Also the rate of successful coalesced memory access depends on the dynamics of the model, i. e. how far each grid point is moved by the advection. Figure 5 shows the entropy for the HMF model for different number of grid points $n_p = n_\theta$. The behavior is qualitatively the same as the previous case.

4.1.2 The Ring model

Computation of the mean-force field (6) for the pair interaction potential given in eq. (2) involves the computation of $\sin(\theta - \theta') = \sin \theta \cos \theta' - \cos \theta \sin \theta'$, and can therefore be optimized by simply computing and storing two arrays with the values of $\sin \theta$ and $\cos \theta$ at the spatial grid points at the beginning of the simulation. The CUDA function *rsqrt* for the inverse of the square-root in eq. (6) is exploited as it is cheaper than to computed both a square root and its inverse. The remaining steps are as described above. The speedups obtained are presented in table 5. The behavior with the number of grid points is similar

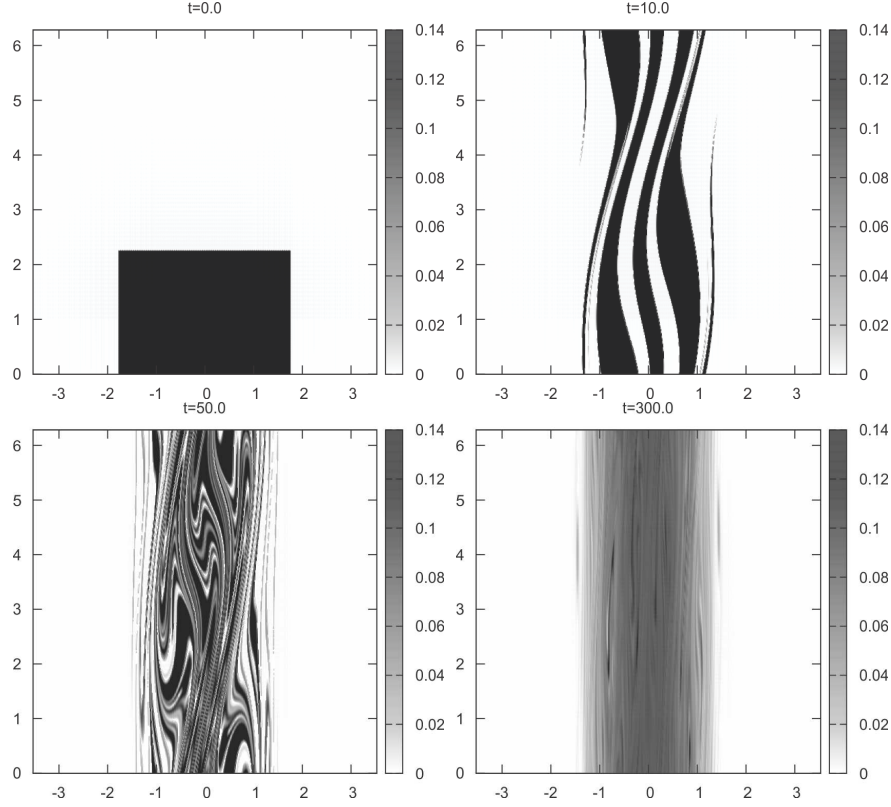


Fig. 3. Snapshots of the distribution function of the HMF model for $t = 0, 10, 50, 300$. In each graphic p and θ correspond to the horizontal and vertical axis respectively.

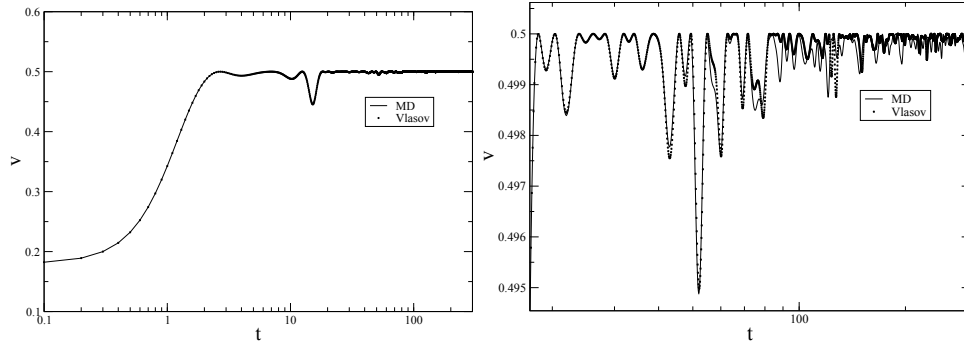


Fig. 4. Mono-Log graphic of the potential energy for the HMF model obtained from Molecular Dynamics simulation (dots) with $N = 20,000,000$ particles and from the numerical solution of the Vlasov equation with $n_p = n_\theta = 2048$ (continuous line). The right panel is a zoom over a region of the left panel.

to the two previous models.

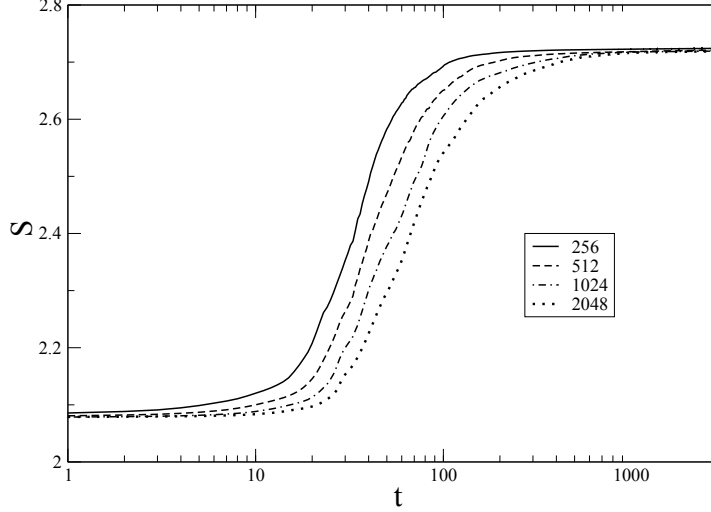


Fig. 5. Entropy for the HMF model for different grid resolutions.

$n_p = n_x$	GTX 570 Ti	GTX 590
256	17	19
512	29	33
1024	46	55
2048	57	73

Table 5

Speedups for the Ring model model.

5 Concluding remarks

We presented a GPU implementation using CUDA of a parallel numeric solver for the Vlasov equation on a GPU, based on a time-split scheme with a modified cubic spline interpolation for both the spatial and momentum direction in phase space. The interpolation relies on a finite-difference scheme to accurately determine the second order derivatives required by the cubic spline interpolation in such a way that only a small number of neighboring points is required, leading to a faster and simpler parallel implementation. Coalesced access to global memory in the GPU is ensured by performing a transpose of the distribution function and its second derivatives when performing the advection in the spatial direction. Implementations for three different one-dimensional long-range interacting models were presented with a discussion of accuracy and speedups of the simulations. The Vlasov dynamics leads to the formation of indentations in a scale which becomes smaller with time, and due to the finite grid accuracy, information loss ensues after some time, leading to a coarse-grained distribution, and an increase in entropy. Before the scale of indentation reaches grid accuracy the entropy is conserved by our approach. The same occurs after the distribution has been coarse-grained.

Higher order time split schemes and a filtering procedure can also be implemented if required [31]. Although there is certainly room for improvements in our algorithm, the speedups obtained allow to conclude that the present parallel implementation is a useful tool in the ongoing investigations on open problems for long-range interacting systems.

6 Acknowledgments

The author would like to thank CNPq and CAPES (Brazil) for partial financial support.

References

- [1] A. Campa, T. Dauxois and S. Ruffo, *Phys. Rep.* **480** (2009) 57.
- [2] *Dynamics and Thermodynamics of Systems with Long-Range Interactions*, T. Dauxois, S. Ruffo, E. Arimondo and M. Wilkens (Eds.), Springer (Berlin, 2002)
- [3] *Dynamics and Thermodynamics of Systems with Long-Range Interactions: Theory and Experiments*, A. Campa, A. Giansanti, G. Morigi and F. S. Labini (Eds.), AIP Conf. Proceedings Vol. 970 (2008).
- [4] *Long-Range Interacting Systems, Les Houches 2008, Session XC*, T. Dauxois, S. Ruffo and L. F. Cugliandolo Eds, Oxford Univ. Press (Oxford, 2010).
- [5] T. M. Rocha Filho, A. Figueiredo and M. A. Amato, *Phys. Rev. Lett.* **95** (2005) 190601.
- [6] A. Figueiredo, T. M. Rocha Filho and M. A. Amato, *Europhys. Lett.* **83** (2008) 30011.
- [7] F. P. C. Benetti, T. N. Teles, R. Pakter and Y. Levin, *cond-mat:1202.1810*.
- [8] T. M. Rocha Filho, M. A. Amato, B. A. Mello and A. Figueiredo, *Phys. Rev. E* **84** (2011) 041121.
- [9] T. Padmanabhan, *Phys. Rep.* **188** (1990) 285.
- [10] Y. Levin, R. Pakter and T. N. Teles, *Phys. Rev. Lett.* **100** (2008) 040604.
- [11] Y. Levin, R. Pakter and T. N. Teles, *Phys. Rev. E* **78** (2008) 021130.
- [12] Y. Sota, O. Iguchi, M. Morikawa, T. Tatekawa and K. I. Maeda, *Phys. Rev. E* **64** (2001) 056133.

- [13] T. Tatekawa, F. Bouchet, T. Dauxois and S. Ruffo, Phys. Rev. E **71** (2005) 056111.
- [14] M. Antoni and S. Ruffo, Phys. Rev. E **52** (1995) 2361.
- [15] T. N. Teles, Y. Levin and R. Pakter, Mon. Not. R. Astron. Soc. **417** (2011) L21.
- [16] M. Joyce and T. Worrakitpoonpon, Phys. Rev. E **84** (2011) 011139.
- [17] K. R. Yawn and B. N. Miller, Phys. Rev. E **68** (2003) 056120.
- [18] R. Bonifacio, F. Casagrande, G. Cerchioni, L. De Salvo Souza, P. Pierini and N. Piovella, Riv. Nuovo Cimento **13** (1990) 1.
- [19] J. L. Tennyson, J. D. Meiss and P. J. Morrison, Physica D **71** (1994) 1.
- [20] W. Braun and K. Hepp, Commun. Math. Phys. **56** (1977) 101.
- [21] A. Antoniazzi, F. Califano, D. Fanelli and S. Ruffo, Phys. Rev. Lett. **98** (2007) 160602.
- [22] P. de Buyl, D. Fanelli and S. Ruffo, cond-mat:1112.1102.
- [23] P. de Buyl, Commun. Nonlinear Sci. Numer. Simulat. **15** (2010) 2133.
- [24] R. Pakter and Y. Levin, Phys. Rev. Lett. **106** (2011) 200603.
- [25] *Space Plasma Simulation*, Lect. Notes in Physics, J. Büchner, C. T. Dum and M. Scholer Eds. Springer (Berlin, 2003).
- [26] E. Pohn, M. Shoucri and G. Kamberlander, Comp. Phys. Comm. **166** (2005) 81.
- [27] N. Crouseilles, M. Gutnic, G. Latu and E. Sonnerdrücker, Comm. Nonlin. Sci. Num. Sim. **13** (2008) 88.
- [28] N. Crouseilles, G. Latu and E. Sonnerdrücker, J. Comp. Phys. **228** (2009) 1429.
- [29] L. K. S. Daldorff and B. Elisson, Parallel Computing **35** (2009) 109.
- [30] E. Bengt, Transp. Theor. Stat. Phys. **39** (2010) 387.
- [31] T. D. Arber and R. G. L. Vann, J. Comp. Phys. **180** (2002) 339.
- [32] F. Filbet and E. Sonnerdrücker, Comp. Phys. Comm. **150** (2003) 247.
- [33] M. Shoucri, Commun. Nonlinear Sci. Numer. Simulat. **13** (2008) 174.
- [34] NVIDIA, CUDA PRogramming Guide, Ver. 4.0, 2011.
- [35] C. Z. Cheng and G. Knorr, J. Comp. Phys. **22** (1976) 330.
- [36] E. Sonnerdrücker, J. Roche, P. Bertrand and A. Ghuizzom J. Comp. Phys. **149** (1999) 201.
- [37] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes* 3rd Ed, Cambridge University Press (Cambridge, 2007).

- [38] G. Latu. *Fine-grained parallelization of Vlasov-Poisson application on GPU*, Euro-Par 2010, Parallel Processing Workshops, Springer (New York, 2011).
- [39] G. Ruetsch and P. Micikevicius, *Optimizing matrix transpose in CUDA*, NVIDIA Technical Report (2009).
- [40] B. Fornberg, Math. Comp. **51** (1988) 699.
- [41] L. Galeotti and F. Califano, Phys. Rev. Lett. **95** (2005) 015002.
- [42] F. Califano and L. Galeotti, Phys. Plasmas **13** (2006) 082102.
- [43] V. Carbone, R. De Marco, F. Valentini and P. Veltri, Eur. Phys. Lett. **78** (2007) 65001.
- [44] H. Yoshida, Phys. Lett. A **150** (1990) 262.